Q1:

a) Manufacturers of ATmega16 claimed that it can achieve up 16 MIPS. What MIPS stands for? When this can occur and how can you explain this claim considering that most of the instructions need 2 cycles for execution?

MIPS= Million Instruction Per Second. This can occur when it operates with 16MHz clock frequency. This is quite true since pipelining is implemented in ATmega16.

b) What are all possible options of ATmega16 clock sources? What are all internal clock values that ATmega16 can work on?

ATmega16 Clock sources: External RC oscillator, External crystal oscillator, External clock, internal RC oscillator.

Internal clock values: 8MHz, 4MHz, 2MHz, 1MHz

c) List three internal interrupt sources and three external interrupt sources in ATmega16.

internal interrupt sources: EEPROM operation(write, read), completion of ADC conversion, timers overflow … (almost all peripheral has interrupt)

external interrupt sources: signals on pins INT0, INT1, T0, T1

d) What are the registers that used for indirect addressing in ATmega16?

X =R27:R26, Y=R29:R28, Z=R31:R30

e) What are the instructions that are used to access (i) I/O space? (ii) data space?

i) IN, OUT   ii) LD, ST

f) How many bits that are actually needed in PC, SP, EEARH registers in ATmega328 if you know that it has 2KB SRAM, 1024 EEPROM, 32KB (16K x 16) Flash memory. What are the addresses of these registers in IO space and in data space?

*PC: 14 bits (PC addresses flash memory which is 16K = 2^4 * 2^10 = 2^14)*

SP: 13 bits (SP addresses SRAM which is: internal SRAM 2K (2048) + I/O space(64) + Register file (32)

EEARH: 2 bits (1024 EEPROM location needs 10 bits for addressing, the lower byte is soured from EEPRL, the rest bits (2 bits) are from EERAH.

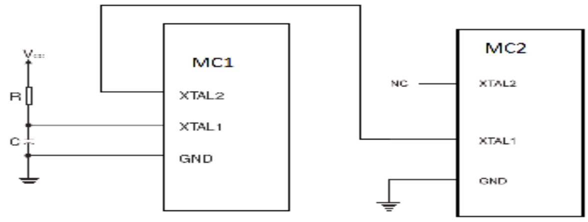
g) If you know that ATmega16 is running with minimum internal clock, what is the minimum possible ADC clock prescaling to digitize: (i) 700Hz signal (ii) 7KHz signal (iii) 700KHz.

We know that sampling rate must not be less than 2*signal frequency. We also know that minimum internal clock of ATmega16 is 1 MHz. Minimum prescaling is given by dividing ATmega16 main clock over the minimum sampling rate

	Signal frequency	Min sampling rate	Min prescling	Closest ATmega16 prescling (<= Min prescaling)
i	700	1400	714	***128*** *since this is the min available prescaling*
ii	7,000	14000	71.4	***64***
iii	700,000	1400,000		***Not applicable*** *since max sps is 15000*

Q2:

a) We have two ATmega16 microcontrollers; MC1 and MC2. We want MC1 be derived with 5MHz external RC oscillator and MC2 be derived with MC1 clock. Show (draw) how to connect XTAL1, XTAL2 for both microcontrollers. Find CKSEL bits for MC1 and MC2.



	CKSEL3	*CKSEL2*	*CKSEL1*	*CKSEL0*	*comment* للتوضيح فقط
MC1	*1*	*0*	*0*	*0*	*external RC oscillator 3 – 8 MHz*
MC2	*0*	*0*	*0*	*0*	*external clock*

b) How many bits that are actually needed in PC, SP, EEARH registers in ATmega644 if you know that it has 4KB SRAM, 2048 EEPROM, 64KB (32K x 16) Flash memory. What are the addresses of these registers in IO space and in data space?

	#BITS	*address (I/O space)*	*address (data space)*	*comment* للتوضيح فقط
PC	*15*	*-*	*-*	*PC addresses flash memory which is 32K (2^5x2^{10}) location*
SP	*13*	*SPH: 3E H* *SPL: 3D H*	*SPH: 5E H* *SPL: 5D H*	*SP addresses stack located in SRAM data space (enternal SRAM (4K)+ I/O space(64) + register file(32)*
EEARH	*3*	*1F H*	*3F H*	*2048 byte EEPROM needs 11 bits to be addressed. EEARH:EERAL addresses EEPROM, the lower 8-bits of the address comes from EERAL, the rest of bits (3bits) come from EERAH*

c) Write instructions to: Add immediate value 78h to the byte in flash memory location 02CDH to internal RAM location 80H, divide by 4 then put the result in RAM memory location where R5:R4 points.

```
LDI     R31, $02            ; store high byte of flash memory location address (02CD H) in ZH
LDI     R30, $0CD           ; store low byte of flash memory location address ($02CD H) in ZL
LPM                         ; load R0 with byte from SRAM location pointed by Z ( R0 ← M[Z])
LDS     R16, $0080          ; load direct from SRAM address 80H
ADD     R16, R0             ; add byte in location 02CDH to internal RAM location 80H
ADI     R16, $78            ; add immediate value 78H
LSL     R16                 ; divide by 2
LSL     R16                 ; divide by 2
MOVW    R27:R26,  R5:R4     ; move address in R5:R4 to X pointer
ST      R16, X              ; store result in SRAM where X points
```

d) Write instructions to: Initialize the stack at the end of the internal RAM of ATmega16 then push the contents of Z pointer into stack.

```
LDI     R17, $5F            ; store high byte address of SRAM end (045F H) in R17
LDI     R16, $04            ; store low byte address of SRAM end (045F  H) in R16
OUT     SPL, R16            ; load SPL with low byte address of SRAM end
OUT     SPH, R17            ; load SPH with high byte address of SRAM end
LPM                         ; contents of Z is now in R0
PUSH    R0                  ; push the contents of Z pointer into stack
```

e) Write instructions to: Make port A, port B as input ports and port C as output port then read port A and port B and output the greater at port C.

```
        CLR     R17             ; store 00H in R17 to make all pins of portA and portB input
        SER     R16             ; store FFH in R16 to make portC output
        OUT     DDRA, R17       ; configure all pins of port A as output
        OUT     DDRB, R17       ; configure all pins of port B as output
        OUT     DDRC, R16       ; configure all pins of port C as input
        IN      R0, PORTA       ; input from port A to R0
        IN      R1, PORTB       ; input from port B to R1
        CMP     R1, R0          ; compare R1 and R0
        BRGE    NEXT            ; if R1 is greater or equal R0 then go to NEXT
        OUT     PORTC, R1       ; otherwise output R1 on port C
        RJMP    END             ; jump to END
NEXT:   OUT     PORTC, R0       ; output R0 on port C
END:
```